OIPE
NOV 18 2002
PATENT & TRADEMARK OFFICE

AF/$
2700
#9
10/13

ATTORNEY DOCKET NO.: W. DIEPSTRATEN 16-2-2 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

RECEIVED
NOV 21 2002
Technology Center 2100

In re Application of:	Wilhelmus J. M. Diepstraten, *et al.*
Serial No.:	09/213,970
Filed:	December 17, 1998
Title:	CONTEXT CONTROLLER HAVING INSTRUCTION-BASED TIME SLICE TASK SWITCHING CAPABILITY AND PROCESSOR EMPLOYING THE SAME
Group:	2154
Examiner:	Larry D. Donaghue

Commissioner for Patents
Washington, D. C. 20231

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Washington, D.C. 20231, on 11/11/2002 (Date)
Stephanie Pritt
(Printed or typed name of person signing the certificate)
[illegible]
(Signature of the person signing the certificate)

Sir:

ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF UNDER 37 C.F.R. §1.192

This is an appeal from a Final Rejection mailed on June 18, 2002, of Claims 1-22. The Appellants submit this Brief in triplicate as required by 37 C.F.R. §1.192(a), with the statutory fee of $300.00 as set forth in 37 C.F.R.§1.17(c), and hereby authorize the Commissioner to charge any

```
11/20/2002 AWONDAF1 00000089 082395    09213970
01 FC:1402          20.00 CH       300.00 OP
```

additional fees connected with this communication or credit any overpayment to Deposit Account No. 08-2395.

This Brief contains these items under the following headings, and in the order set forth below in accordance with 37 C.F.R. §1.192(c):

I. REAL PARTY IN INTEREST

The real party in interest in this appeal are the joint Assignees Agere Systems, Incorporated and Choice-Intersil Microsystems, Incorporated.

II. RELATED APPEALS AND INTERFERENCES

No other appeals or interferences will directly affect, be directly affected by, or have a bearing on the Board's decision in this appeal.

III. STATUS OF THE CLAIMS

Claims 1-22 are pending in this Application.

IV. STATUS OF THE AMENDMENTS

The present Application was filed on December 17, 1998, containing Claims 1-22. The Appellants filed a first Amendment on March 14, 2002, in response to an Examiner's Action mailed February 12, 2002. The first Amendment amended Claims 1, 8 and 15. No claims were canceled or added. In a Final Rejection mailed on June 18, 2002, the Examiner indicated that the first Amendment did not place the Application in condition for allowance. In response, the Appellants filed a first Request for Reconsideration on July 22, 2002. The Examiner indicated in an Advisory

Action mailed on August 20, 2002 that the Request for Reconsideration filed on July 22, 2002, did not place the Application in condition for allowance. The Examiner also indicated that upon filing an appeal that the first Amendment would be entered. The Appellants then filed a Notice of Appeal on September 18, 2002.

V. SUMMARY OF THE INVENTION

The present invention is directed, in general, to computer processors and, more specifically, to a context controller having instruction-based time slice task switching capability and a processor employing the context controller. (Page 2, lines 11-14). The present invention provides a context controller for managing multitasking in a processor and a method of operating the same. (Page 9, lines 1-4).

In one embodiment, the context controller includes: (1) a time slice instruction counter that counts a number of instructions executed with respect to a given background task and (2) a background task controller that cyclicly executes a context corresponding to another background task when the number of instructions executed equals a dynamically-programmable time slice value. (Page 9, lines 4-10).

The present invention therefore introduces the broad concept of providing variable instruction-based time slice (which may be thought of as "instruction slice") multitasking in which the time slice value (the number of instructions to be executed with respect to each background task in its allotted time slice) remains fully programmable during execution of the background tasks. For purposes of the present invention, "dynamically-programmable" is defined as being programmable subsequent to system initialization. Additionally, "context" is defined as all processor state information (or any subset thereof, and such as register values) that would be of use in restoring the processor to a given state. (Page 9, lines 11-22).

IN THE ENVIRONMENT OF A PREFERRED EMBODIMENT

An embodiment of the present invention is illustrated in FIGURE 5 showing an interaction diagram of an internal structure of the context controller illustrated in FIGURE 4. (Page 13, lines 18-20). Two processes are illustrated as being contained in the context controller block 110. An event synchronizer 150 accepts exogenous event signals from an AsyncEvents signal route 158 and synchronizes them with the master clock rising edge Mr 517, which is provided by the clock generator 106 via a ClkSyn signal route 156. These events are sent on, via a SyncEvents signal route 166, as event signals, just as with the (inherently synchronized) event signals from endogenous sources on a PriDP signal route 164. (Page 42 line 19 to Page 43, line 2).

The fundamental context control state machine operates in an event prioritizer 152 which receives input signals from the clock generator 106 via a ClkPri signal route 154, event signals from the event-synchronizer 150 via a SyncEvents signal route 166 and data path CctlDP functions 143 via a PriDP signal route 164. Additionally, decode signals for various instructions relevant to context control and inter-context communication from an instruction decoder over the InstCctl channel 142 via an InstPri signal route 162 are received. (Page 43, lines 3-12).

VI. ISSUES

A. First Issue Presented for Review:

Whether Claims 1, 4, 7, 8, 11 and 14 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 5,528,513 to Vaitzblit, *et al.* ("Vaitzblit") in view of U.S. Patent No. 6,009,454 to Dummermuth, *et al.* ("Dummermuth").

B. Second Issue Presented for Review:

Whether Claims 2, 6, 9 and 13 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and Dummermuth in view of U.S. Patent No. 6,085,218 to Carmon.

C. Third Issue Presented for Review:

Whether Claims 3 and 10 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and Dummermuth in view of U.S. Patent No. 5,239,652 to Seibert, *et al.* ("Seibert").

D. Fourth Issue Presented for Review:

Whether Claims 15, 18, 19 and 21-22 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and Dummermuth in further view of U.S. Patent No. 5,713,038 to Motomura.

E. Fifth Issue Presented for Review:

Whether Claim 17 is properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit, Dummermuth and Motomura and further in view of Seibert.

F. Sixth Issue Presented for Review:

Whether Claims 16 and 20 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit, Dummermuth and Motomura and in further view of Carmon

VII. GROUPING OF THE CLAIMS

Claims 1-7, 8-14 and 15-22 do not stand or fall together. Claims 1-7 form a first group and are patentably distinct from Claims 8-14 and Claims 15-22. Claims 8-14 form a second group and are likewise patentably distinct from Claims 1-7 and Claims 15-22. Finally, Claims 15-22 form a third group and are patentably distinct from Claims 1-7 and Claims 8-14.

VIII. PRIOR ART

A. Vaitzblit

Vaitzblit is directed to a scheduler in a continuous media file server that supports multiple classes of tasks, isochronous, real-time and general-purpose, with diverse performance requirements. (Abstract). The general-purpose class supports preemptive tasks that are suitable for low-priority background processing. (Column 3, lines 33-34). The general-purpose tasks are executed in a round-robin fashion and run until blocked or a quantum timer expires. (Column 5, lines 16-25).

B. Dummermuth

Dummermuth is directed to a multi-tasking operating system for real-time control of industrial processes. (Abstract). Multi-tasking is provided by recognizing that both ladder-type and state-type programs can be considered as simply a collection of individual instructions linked together by an implicit pointer list. At the conclusion of any instructions, a pointer may be developed to a single next instruction. (Column 2, lines 48-53). The multi-tasking operation switches between tasks after a predetermined number of instructions by making the execution of each instruction explicit. (Column 7, lines 23-26). This differs from the present invention where a slice value is dynamically-programmable instead of predetermined.

C. Carmon

Carmon is directed to monitoring a multi-task system by detecting overruns of any task beyond a declared maximum processor cycle limit for the task. Counter hardware and control logic reduces software overhead for monitoring execution cycle utilization by a task and provides capability not only of overrun detection, but programmed cycle usage alarm, consumed cycle count and overall processor loading or utilization measurements to be made. (Abstract). Carmon presumes that each task in a multi-tasking system will be held to the requirement of declaring its own maximum execution cycle count limit that will be required by the task running to completion. Carmon accurately counts only processor cycles used by each individual task or program during its execution and will not count those machine cycles during the execution time of a task within which code is not actually being executed. (Column 3, lines 6-11).

D. Seibert

Seibert is directed to reducing the power consumption of a computer by determining when the central processing unit (CPU) is not actively processing and generating a power-off signal to a control logic circuit. (Abstract). The control logic circuit controls the CPU, which is powered by a power supply, such that the CPU is completely disconnected from the power supply and brought into substantially full power periodically at predetermined intervals by interrupts when the CPU is in an inactive state. (Column 3, lines 27-35).

E. Motomura

Motomura is directed to a microprocessor that has a register file which allows a higher speed, more flexible, context switching as compared to conventional microprocessors. (Column 3, lines 15-17). The microprocessor is coupled to a memory and includes an instruction pipeline and a register file. At the time of context switching, the overhead required for the save/restore of the content stored in the register file can be reduced. (Abstract).

IX. THE APPELLANTS' ARGUMENTS

The inventions set forth in independent Claims 1, 8 and 15 and their respective dependent claims are not obvious over the references on which the Examiner relies.

A. Rejection of Claims 1, 4, 5, 7, 8, 11, 12 and 14 under 35 U.S.C. §103

The Examiner rejected Claims 1, 4, 7, 8, 11 and 14 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of Dummermuth. The Examiner asserts that Vaitzblit teaches the invention substantially as claimed including a background controller operating in a cyclical manner. The Appellants respectfully disagree.

In Vaitzblit, there may be many general-purpose class tasks in a server 20, all of which need to make progress in order for the server 20 to function. The general-purpose tasks, therefore, are granted a minimum CPU processing quantum to ensure that general-purpose tasks can always make progress. (Column 3, lines 33-43). The general-purpose tasks that are ready for execution are placed on a general-purpose ready queue 108, which is served in a round-robin fashion. If the general-purpose ready queue 108 is not empty, the scheduler 53 starts a general-purpose quantum timer, and activates the first task from the general-purpose ready queue 108. (Column 5, lines 15-22).

The first task in the general-purpose ready queue runs until it blocks or the quantum timer expires. If the task blocks, its context is saved on a wait queue and the first task from the general-purpose ready queue is restored for execution. If the quantum timer expires, the scheduler 53 saves the context of the currently running task at the end of the GP ready queue and switches to a new round of servicing the real-time tasks. The execution of the general-purpose tasks may be preempted one or more times by the isochronous tasks. After each preemption, the execution of the general-purpose class continues until the total time spent in processing general-purpose tasks reaches the guaranteed quantum. (Column 5, lines 22-32).

The scheduler, however, does not cyclicly activate a context corresponding to another background task when the number of executed instructions of a given background task equals a dynamically-programmable time slice value. (Claims 1 and 8). Instead, the scheduler processes another general-purpose task after processing of a first general-purpose task is complete and the quantum timer has not expired. Additionally, the scheduler will switch from processing a general-purpose task to processing an isochronous task when preempted. Finally, the scheduler will switch to processing a real-time task when the quantum timer expires. Vaitzblit, therefore, teaches switching between general-purpose tasks when processing of the first task is complete (block) and the quantum timer has not expired. Vaitzblit does not teach or suggest, however, cyclicly activating a context corresponding to another background task when the number of instructions executed with respect to a given background task equals a dynamically-programmable time slice value as recited in independent Claims 1 and 8.

The Examiner also asserts that Dummermuth teaches a controller that cyclicly activates context according to a number of instruction executed and a counter that counts the number of executed instructions with respect to a given task. (Examiner's Action mailed on June 18, 2002, page 2). Dummermuth, however, does not teach a context controller that cyclicly activates a context corresponding to a background task when the number of instructions executed with respect to another background task equals a dynamically-programmable time slice value. (Claims 1 and 8).

Dummermuth discloses a multi-tasking operating system for real-time control of industrial processes. (Abstract). Multi-tasking is provided by recognizing that both ladder-type and state-type programs can be considered as simply a collection of individual instructions linked together by an implicit pointer list. At the conclusion of any instructions, a pointer may be developed to a single next instruction. (Column 2, lines 48-53). The operating system "switches between tasks after a predetermined number of instructions by making the execution of each instruction explicit." (Column 7, lines 23-26). This differs from the present invention where a slice value is dynamically-programmable instead of predetermined.

Since neither Vaitzblit nor Dummermuth teach or suggest cyclicly activating a context corresponding to another background task when the number of instructions executed with respect to a given background task equals a dynamically-programmable time slice value as recited in independent Claims 1 and 8, the Examiner cannot establish a *prima facie* case of obviousness of Claims 1 and 8 and Claims dependent thereon. Accordingly, the Appellants respectfully traverse the Examiner's rejection of Claims 1, 4, 5, 7, 8, 11, 12 and 14 under 35 U.S.C. §103(a) and request a reversal of the Examiner's rejection with respect to these claims.

Furthermore, the combination of Vaitzblit and Dummermuth is improper. As the board understands, obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is a teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. *In re Fine*, 5 U.S.P.Q. 2d 1596 (1998). The motivation to combine references may ". . . come from the nature of a problem to be solved, leading inventors to look for references relating to possible solutions to that problem." *Pro-Mold and Tool Company v. Great*

Lakes Plastics Inc. 37 U.S.P.Q.2d 1626 (1996). However, hindsight is never appropriate motivation for combining references. To this end, relying upon hindsight knowledge of the Appellant's disclosure when the prior art does not teach nor suggest such knowledge, results in the use of the invention as a template for its own reconstruction. This is inappropriate in the determination of patentability. *Sensonics Inc. v. Garlock, Inc.* 220 U.S.P.Q. 303 (1983).

One skilled in the art would not be motivated to combine Vaitzblit with Dummermuth to arrive at the claimed invention because Vaitzblit is directed to a scheduler for a continuous media file server and Dummermuth is directed to a multi-tasking operating system for real-time control of industrial processes. More specifically, Vaitzblit is directed to solving the problem of scheduling multiple classes of tasks with diverse performance requirements for use in continuous media applications and Dummermuth is directed to solving the multi-tasking of special purpose computers uniquely suited for the industrial controller environment.

B. Rejection of Claims 2, 6, 9 and 13 under 35 U.S.C. §103

The Examiner rejected Claims 2, 6, 9 and 13 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and Dummermuth in view of Carmon. Neither Vaitzblit nor Dummermuth, however, individually or in combination, teach or suggest each and every element of independent Claims 1 and 8. Furthermore, Carmon does not cure the deficiencies of Vaitzblit and Dummermuth. Instead, Carmon is directed to monitoring a multi-task system by detecting overrun of any task beyond a declared maximum processor cycle limit for the task. (Abstract). Since the Examiner did not establish a *prima facie* case of obviousness of independent Claims 1 and 8, and Carmon does not cure the deficiencies of Vaitzblit and Dummermuth, then the Examiner cannot establish a case of obviousness of dependent Claims 2, 6, 9 and 13. Accordingly, the Applicants respectfully traverse the Examiner's rejection of Claims 2, 6, 9 and 13 under 35 U.S.C. §103(a) and request a reversal of the Examiner's rejection with respect to these claims.

C. Rejection of Claims 3 and 10 under 35 U.S.C. §103

The Examiner rejected Claims 3 and 10 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and Dummermuth in view of Seibert. Neither Vaitzblit nor Dummermuth, however, individually or in combination, teach or suggest each and every element of independent Claims 1 and 8. Furthermore, Seibert does not cure the deficiencies of Vaitzblit and Dummermuth. Instead, Seibert is directed to reducing the power consumption of a computer by determining when the central processing unit is not actively processing and generating a power-off signal to a control logic circuit. (Abstract). Since the Examiner did not establish a *prima facie* case of obviousness of independent Claims 1 and 8, and Seibert does not cure the deficiencies of Vaitzblit and Dummermuth, then the Examiner cannot establish a case of obviousness of dependent Claims 3 and 10. Accordingly, the Applicants respectfully traverse the Examiner's rejection of Claims 3 and 10 under 35 U.S.C. §103(a) and request a reversal of the Examiner's rejection with respect to these claims.

D. Rejection of Claims 15, 18, 19 and 21-22 under 35 U.S.C. §103

The Examiner rejected Claims 15, 18, 19 and 21-22 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and Dummermuth in further view of Motomura. As discussed above with respect to independent Claims 1 and 8, Vaitzblit and Dummermuth do not teach or suggest switching between background tasks using a dynamically-programmable time slice value as claimed in independent Claim 15. Furthermore, Motomura does not cure the deficiencies of Vaitzblit and Dummermuth. Instead, Motomura is directed to a microprocessor that has a register file which allows a higher speed, more flexible, context switching as compared to conventional microprocessors. (Column 3, lines 15-17).

Since Vaitzblit and Dummermuth do not teach each and every element of independent Claim 15 and Motomura does not cure its deficiencies, then the Examiner can not establish a *prima facie* case of obviousness of independent Claim 15 and Claims dependent thereon. Accordingly, the Applicants respectfully traverse the Examiner's rejection of Claims 15, 18, 19 and 21-22 under 35 U.S.C. §103(a) and request a reversal of the Examiner's rejection with respect to these claims. Furthermore, a person having ordinary skill in the art could not have arrived at the processor claimed in Claim 15 by combining the teachings of Vaitzblit, Dummermut and Motomura since neither Vaitzblit, Dummermut or Motomura teach or suggest switching between tasks using a dynamically-programmable time slice value.

E. Rejection of Claim 17 under 35 U.S.C. §103

The Examiner rejected Claim 17 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit, Dummermuth and Motomura and further in view of Seibert. As stated above, Vaitzblit and Dummermuth do not teach or suggest each and every element of independent Claim 15. Furthermore, neither Motomura or Seibert cure the deficiencies of Vaitzblit and Dummermuth. No combination, therefore, of Motomura, Seibert, Vaitzblit, or Dummermuth teaches or suggests each and every element of Claim 17. The Examiner, nevertheless, cannot establish a *prima facie* case of obviousness with respect to Claim 17 which includes each element of independent Claim 15. Accordingly, the Applicants respectfully traverse the Examiner's rejection of Claim 17 under 35 U.S.C. §103(a) and request a reversal of the Examiner's rejection with respect to these claims.

F. Rejection of Claims 16 and 20 under 35 U.S.C. §103

The Examiner rejected Claims 16 and 20 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit, Dummermuth and Motomura and in further view of Carmon. Neither Vaitzblit nor Dummermuth teach or suggest each and every element of independent Claim 15. Furthermore, neither Motomura or Carmon cure the deficiencies of Vaitzblit and Dummermuth. No combination, therefore, of Motomura, Vaitzblit, Carmon or Dummermuth teaches or suggests each and every element of Claims 16 and 20. The Examiner, nevertheless, cannot establish a *prima facie* case of obviousness with respect to Claims 16 and 20. Accordingly, the Applicants respectfully traverse the Examiner's rejection of Claims 16 and 20 under 35 U.S.C. §103(a) and request a reversal of the Examiner's rejection with respect to these claims.

In view of the foregoing remarks, the cited references do not support the Examiner's rejection of Claims 1-22 under 35 U.S.C. §103(a). Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection of all of the Appellants' pending claims.

Respectfully submitted,

Hitt Gaines & Boisbrun, P.C.

[illegible]

David H. Hitt
Registration No. 33,182

Dated: November 11, 2002

Hitt Gaines & Boisbrun, P.C.
P.O. Box 832570
Richardson, Texas 75083
(972) 480-8800
(972) 480-8865
E-Mail: dhitt@abstractassets.com

X. APPENDIX A - CLAIMS

1. (Amended) A context controller for managing multitasking of a plurality of tasks including foreground tasks and background tasks in a processor, comprising:

a time slice instruction counter that counts a number of instructions executed with respect to a given background task; and

a background task controller that cyclicly activates a context corresponding to another background task when said number equals a dynamically-programmable time slice value.

2. The context controller as recited in Claim 1 wherein said time slice instruction counter initially contains said dynamically-programmable time slice value as a time slice for said given background task begins, said time slice instruction counter decrementing as said instructions with respect to said given background task are executed.

3. The context controller as recited in Claim 1 wherein said context controller places said processor in an idle state when all of said background tasks are inactive.

4. The context controller as recited in Claim 1 wherein said background task controller is adapted to activate a context corresponding to a particular background task by vectoring to a software-selectable memory location.

5. The context controller as recited in Claim 1 further comprising a foreground task controller that activates contexts corresponding to foreground tasks based on priority and in response to events, said background task controller cyclicly activating contexts corresponding to said background tasks subject to activation of said contexts corresponding to said foreground tasks.

6. The context controller as recited in Claim 1 wherein said dynamically-programmable time slice value is contained in a register of said processor.

7. The context controller as recited in Claim 1 wherein application tasks executing on said processor can program said dynamically-programmable time slice value.

8. (Amended) A method of managing multitasking of a plurality of tasks including foreground tasks and background tasks in a processor, comprising the steps of:

counting a number of instructions executed with respect to a given background task; and

cyclicly activating a context corresponding to another background task when said number equals a dynamically-programmable time slice value.

9. The method as recited in Claim 8 wherein said step of counting comprises the steps of:

initializing a time slice instruction counter with said dynamically-programmable time slice value as a time slice for said given background task begins; and

decrementing said time slice instruction counter as said instructions with respect to said given background task are executed.

10. The method as recited in Claim 8 further comprising the step of placing said processor in an idle state when all of said background tasks are inactive.

11. The method as recited in Claim 8 wherein said step of cyclically activating comprises the step of vectoring to a software-selectable memory location.

12. The method as recited in Claim 8 further comprising the step of activating contexts corresponding to foreground tasks based on priority and in response to events, said step of cyclically

activating comprising the step of cyclicly activating contexts corresponding to said background tasks subject to activation of said contexts corresponding to said foreground tasks.

13. The method as recited in Claim 8 further comprising the step of storing said dynamically-programmable time slice value in a register of said processor.

14. The method as recited in Claim 8 further comprising the step of programing said dynamically-programmable time slice value with application tasks executing on said processor.

15. (Amended) A processor, comprising:

an instruction decoder that decodes instructions received into said processor and corresponding to a plurality of tasks which includes foreground tasks and background tasks;

a plurality of register sets, corresponding to said plurality of tasks, that contain operands to be manipulated;

an execution core, coupled to said instruction decoder and said plurality of register sets, that executes instructions corresponding to an active one of said plurality of tasks to manipulate ones of said operands; and

a context controller, coupled to said instruction decoder and said execution core, that manages multitasking with respect to said plurality of tasks, including:

a time slice instruction counter that counts a number of instructions executed with respect to a given background task; and

a background task controller that cyclicly activates a context corresponding to another background task when said number equals a dynamically-programmable time slice value.

16. The processor as recited in Claim 15 wherein said time slice instruction counter initially contains said dynamically-programmable time slice value as a time slice for said given background task begins, said time slice instruction counter decrementing as said instructions with respect to said given background task are executed.

17. The processor as recited in Claim 15 wherein said context controller places said processor in an idle state when all of said background tasks are inactive.

18. The processor as recited in Claim 15 wherein said background task controller is adapted to activate a context corresponding to a particular background task by vectoring to a software-selectable memory location.

19. The processor as recited in Claim 15 wherein said context controller further includes a foreground task controller that activates contexts corresponding to foreground tasks based on priority and in response to events, said background task controller cyclicly activating contexts corresponding to said background tasks subject to activation of said contexts corresponding to said foreground tasks.

20. The processor as recited in Claim 15 wherein said dynamically-programmable time slice value is contained in a register of said processor.

21. The processor as recited in Claim 15 wherein application tasks executing on said processor can program said dynamically-programmable time slice value.

22. The processor as recited in Claim 15 wherein said processor forms a portion of a general-purpose computer.